Requirements Engineering 4: Software Requirements Specifications

Steve Easterbrook

9/17/97

Outline

Functional Requirements Specifications

Non-Functional Requirements

Qualities of an SRS

IEEE & DOD documentation standards

Concept of Operations Documents

Software Requirements Specification

- **The SRS is the main output of the Requirements Engineering Process**
- **Purpose**
  - **Communicates an understanding of the application domain and the system (machine) to be developed**
  - **Contractual**
  - **Baseline for evaluating subsequent products**
  - **Baseline for change control**
- **Audience**
- **Contents**
- **Structure**

Source: *Adapted from Loucopoulos & Karakostas, 1995, p9*

Functional vs. Non-functional

- **SRS normally specifies "Functional Requirements"**
  - **fundamental functions of the system**
  - **E.g. mapping of inputs to outputs**
  - **E.g. control sequencing**
  - **E.g. timing of functions**
  - **E.g. handling of exceptional situations**
  - **E.g. formats of input and output data (and stored data?)**
  - **E.g. real world entities and relationships modeled by the system**
- **plus some "Non-Functional Requirements (NFRs)"**
  - **constraints on the system**
  - **E.g. security, safety, availability, usability, performance, portability,…**
  - **constraints from the application domain**
  - **E.g. compatibility with (and reuse of) legacy systems**

Source: *Adapted from Loucopoulos & Karakostas, 1995, p10-12*

SRS should not include

- **Project development plans (cost, staffing, schedules, methods, tools, etc)**
  - **Lifetime of SRS is until end of operations phase**
  - **Lifetime of development plans is much shorter**
- **Product assurance plans (CM, V&V, test, QA, etc)**
  - **Different audiences**
  - **Different lifetimes**
- **Designs**
  - **Requirements and designs have different audiences**
  - **Analysis and design are different areas of expertise (I.e. requirements analysts shouldn't do design!)**
  - ***Except where application domain constrains the design: e.g. limited communication between different subsystems for security reasons.***

Source: *Adapted fromDavis, 1990, p183*

On the dominance of FRs

- **To much detail for stakeholders and analysts**
  - **emphasis on functional SRS forces premature fixing of the system boundary before really understanding the application domain**
  - **no clear way to distinguish early architectural decisions (mixed in with FRs)**
- **Obscures other important aspects**
  - **objectives of the system & relationship to enterprise goals**
  - **de-emphasizes the feasibility study**
  - **prior constraints on development (tools, expertise, economics, etc)**
- **Ignores need for negotiation**
  - **No room for identifying and resolving conflict among stakeholders**
  - **Makes it hard to prioritize requirements**
  - **Makes it hard to evaluate alternative ways of achieving the organizational purpose**

Source: *Adapted from Loucopoulos & Karakostas, 1995, p10*

And just what is a NFR?

- **The distinction between functional and non-functional requirements is fuzzy**
  - **…and probably not a very helpful distinction to make**
- **NFRs tend to refer to systemic properties**
  - **But may get allocated to subsystems as design progresses**
  - **E.g. Security**
- **Hence:**
  - **Worry not whether you're specifying functional or non-functional requirements**
  - **But do make sure that they are all *specified behaviorally* (I.e. there is some procedure specified for determining whether they have been met)**
  - **Requirements that are specified non-behaviorally are very difficult to measure.**

Source: *Adapted from Loucopoulos & Karakostas, 1995, p12, and Weiringa, 1996, p21*

A complication: procurement

- **SRS may be written by procurer**
  - **Is really a call for proposals**
  - **Must be general enough to yield a good selection of bids…**
  - **…and specific enough to exclude unreasonable bids**
- **SRS may be written by the bidders**
  - **Represents a proposal to implement a system to meet the CfP**
  - **must be specific enough to demonstrate feasibility and technical competence**
  - **…and general enough to avoid over-commitment**
- **SRS may be written by the selected developer**
  - **reflects the developer's understanding of the customers needs**
  - **forms the basis for evaluation of contractual performance**
- **IEEE Standard recommends SRS jointly developed by procurer and developer**

Attributes of the perfect SRS

- **Valid (or 'correct')**
  - **expresses only actual requirements**
- **Complete**
  - **Specifies all the things the system must do**
  - **...and all the things it must not do!**
  - **Responses to all classes of input**
  - **Structural completeness, and no TBDs!!**
- **Consistent**
  - **doesn't contradict itself (I.e. is satisfiable)**
  - **Uses all terms consistently**
  - **Note: timing and logic are especially prone to inconsistency**
- **Necessary**
  - **doesn't contain anything that isn't "required"**
- **Unambiguous**
  - **every statement can be read in exactly one way**
  - **define confusing terms in a glossary**
- **Verifiable**
  - **a process exists to test satisfaction of each requirement**
  - ***"every requirement is specified behaviorally"***
- **Understandable**
  - **by non-computer specialists**

Source: *Adapted from Davis, 1990, p184-191 and the IEEE-STD-830-1993*

But a perfect specification is unattainable...

Ambiguity Test

- **"The system shall report to the operator all faults that originate in critical functions or that occur during execution of a critical sequence and for which there is no fault recovery response."**

Originate in critical functions	**F**	**T**	**F**	**T**	**F**	**T**	**F**	**T**
Occur during critical seqeunce	**F**	**F**	**T**	**T**	**F**	**F**	**T**	**T**
No fault recovery response	**F**	**F**	**F**	**F**	**T**	**T**	**T**	**T**
Report to operator?								

Source: *Adapted from Easterbrook & Callahan, 1997.*

Avoiding ambiguity

- **Review natural language specs for ambiguity**
  - **use people with different backgrounds**
  - **include software people, domain specialists and user communities**
  - **Must be an independent review (I.e. not by the authors!)**
- **Use a specification language**
  - **E.g. a restricted subset of English**
  - **E.g. a semi-formal notation (graphical, tabular, etc)**
  - **E.g. a formal specification language (e.g. Z, VDM, SCR, …)**
- **Exploit redundancy**
  - **Restate a requirement to help the reader confirm her understanding**
  - **...but clearly indicate the redundancy**
  - **May want to use a more formal notation for the re-statement**

TBDs

- **A specification is not complete if it contains TBDs *("To be determined")***
- **However, TBDs may be necessary as a specification evolves**
- **Every TBD should be accompanied by:**
  - **the reason for the TBD (I.e. why is the information not yet available)**
  - **an indication of how to resolve the TBD**
  - **an indication of who is responsible for resolving it**
  - **a date by when it should be resolved.**
- ***If you don't include this when you write the TBD, you'll never remember it.***

Source: *Adapted from Davis, 1990, p190*

SRS format and style

- **Modifiability**
  - **well-structured, indexed, cross-referenced, etc.**
  - **redundancy should be avoided or must be clearly marked as such**
  - **An SRS is not modifiable if it is not traceable...**
- **Traceability**
  - **Backwards: each requirement traces to a source (e.g. a requirement in the system spec; a stakeholder; etc)**
  - **Forwards: each requirement traces to parts of the design that satisfy that requirement**
  - **Note: traceability links are two-way; hence other documents must trace into the SRS**
  - **Hence every requirement must have a unique label.**
- **Useful Annotations**
  - **E.g. relative necessity and relative stability**

Source: *Adapted from Davis, 1990, p192-5*

Typical Structure (IEEE)

- **1 Introduction**
  - **Purpose**
  - **Scope**
  - **Definitions, acronyms, abbreviations**
  - **Reference documents**
  - **Overview**
- **2 Overall Description**
  - **Product perspective**
  - **Product functions**
  - **User characteristics**
  - **Constraints**
  - **Assumptions and Dependencies**
- **3 Specific Requirements**
- **Appendices**
- **Index**

Identifies the product, & application domain

Describes contents and structure of the remainder of the SRS

Describes all external interfaces: system, user, hardware, software; also operations and site adaptation, and hardware constraints

Summary of major functions

Anything that will limit the developer's options (e.g. regulations, reliability, criticality, hardware limitations, parallelism, etc)

All the requirements go in here (I.e. this is the body of the document). IEEE STD provides 8 different templates for this section

Source: *Adapted from IEEE-STD-830-1993*

IEEE STD Section 3 (example)

3.1 External Interface Requirements
3.1.1 User Interfaces
3.1.2 Hardware Interfaces
3.1.3 Software Interfaces
3.1.4 Communication Interfaces

3.2 Functional Requirements
this section organized by mode, user class, feature, etc. For example:
3.2.1 Mode 1
3.2.1.1 Functional Requirement 1.1
…
3.2.2 Mode 2
3.2.1.1 Functional Requirement 1.1
…
...
3.2.2 Mode n
...

3.3 Performance Requirements
Remember to state this in measurable terms!

3.4 Design Constraints
3.4.1 Standards compliance
3.4.2 Hardware limitations
etc.

3.5 Software System Attributes
3.5.1 Reliability
3.5.2 Availability
3.5.3 Security
3.5.4 Maintainability
3.5.5 Portability

3.6 Other Requirements

Source: *Adapted from IEEE-STD-830-1993*

MIL-STD-498

- **MIL-STD-498 is the main DOD standard for software development and documentation**
  - **replaces DOD-STD-2167A and DOD-STD7935A**
- **Consists of:**
  - **a guidebook,**
  - **a list of process requirements**
  - **22 Data Items Descriptions (DIDs)**
- **DIDs are the documents produced during software development. e.g.**
  - **OCD - Operational Concept Description**
  - **SSS - System/Subsystem Specification**
  - **SRS - Software Requirements Specification**
  - **IRS - Interface Requirements Specification**
  - **etc**

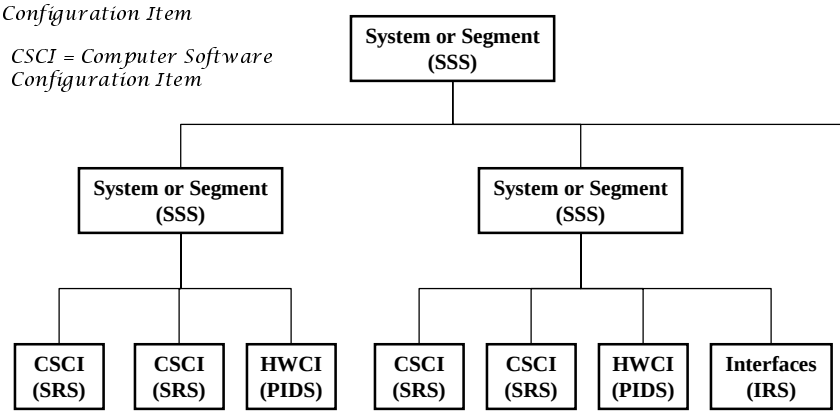
Source: *Adapted from MIL-STD-498*

System Structure

- **MIL-STD-498 uses the following system structure:**

HWCI = Hardware Configuration Item

CSCI = Computer Software Configuration Item



Source: *Adapted from MIL-STD-498*

SRS DID from MIL-STD-498

1 Scope
- **1.1 Identification**
- **1.2 System Overview**
- **1.3 Document Overview**

2 Referenced Documents

3 Requirements
- **3.1 Required States and Modes**
- **3.2 CSCI Capability Requirements**
  - *3.2.x Capability X…*
- **3.3 CSCI External Interface Requirements**
  - 3.3.1 Interface Identification and diagrams
  - 3.3.x Project Unique Identifier
- **3.4 CSCI Internal Interface Requirements**
- **3.5 CSCI Internal Data Requirements**
- **3.6 Adaptation Requirements**
- **3.7 Safety Requirements**
- **3.8 Security and Privacy Requirements**
- **3.9 CSCI Environment Requirements**
- **3.10 Computer Resource Requirements**
- **3.11 Software Quality Factors**
- **3.12 Design and Implementation Constraints**
- **3.13 Personnel-related Requirements**
- **3.14 Training-related Requirements**
- **3.15 Logistics-related Requirements**
- **3.16 Other Requirements**
- **3.17 Packaging Requirements**
- **3.18 Precedence and criticality of Requirements**

4 Qualification Provisions

5 Requirements Traceability

6 Notes

Appendices

Source: *Adapted from MIL-STD-498*

Overcoming functional bias

- **Review (with users, purchasers, etc)**
  - **Should not proceed to development with out it!**
  - **But can turn into a 'dog and pony show'**
  - **Users and buyers overwhelmed by technical detail**
- **Draft users manual**
  - **Helps show that developer understands users' needs.**
  - **But hard to keep current, and hard to trace to specifications**
- **Prototyping**
  - **Helps to pin down user requirements**
  - **But may mislead users, and may freeze the design prematurely**
- **Concept of Operations Document**
  - **A bridge between the user needs and the requirements specification**

Source: *Adapted from Fairley and Thayer, 1997, p73-4*

Concept Analysis Process

- **Analysis of a problem domain and an operational environment to specify characteristics of a proposed system**
  - **This is a systems level approach**
  - **Emphasizes integrated view of the entire system**
  - **Surfaces and prioritizes differing needs**
  - **Helps clarify and resolve conflicts**
- **The ConOps document**
  - **written in narrative prose in the language of the (users) application domain**
  - **Needs don't need to be quantified**
  - **Level of detail can be tailored to the specific situation**
  - **Can use storyboards, informal diagrams, etc**

Source: *Adapted from Fairley and Thayer, 1997, p76*

OCD DID from MIL-STD-498

1 Scope
- **1.1 Identification**
- **1.2 System Overview**
- **1.3 Document Overview**

2 Referenced Documents

3 Current system or situation
- **3.1 Background, objectives and scope**
- **3.2 Operational Policies and constraints**
- **3.3 Description of current system or situation**
- **3.4 Users or involved personnel**
- **3.5 Support concept**

4 Justification for and nature of changes
- **4.1 Justification for change**
- **4.2 Description of needed changes**
- **4.3 Priorities among the changes**
- **4.4 Changes considered but not included**
- **4.5 Assumptions and constraints**

5 Concept for a new or modified system
- **5.1 Background, objectives and scope**
- **5.2 Operational Policies and constraints**
- **5.3 Description of new or modified system**
- **5.4 Users / affected personnel**
- **5.5 Support concept**

6 Operational Scenarios

7 Summary of Impacts
- **7.1 Operational Impacts**
- **7.2 Organizational Impacts**
- **7.3 Impacts during development**

8 Analysis of the Proposed System
- **8.1 Summary of advantages**
- **8.2 Summary of disadvantages/limitations**
- **8.3 Alternatives and trade-offs considered**

Source: *Adapted from MIL-STD-498*

Next Week

- **Requirements Elicitation**
- **Ethnographic Techniques**
- **Scenarios (use-cases)**
- **Formal Inspection exercise (really!)**

References

- Loucopoulos, P. and Karakostas, V. “System Requirements Engineering”. McGraw Hill, 1995.
- Davis, A. M. “Software Requirements: Analysis and Specification”. Prentice-Hall, 1990.
- Wieringa, R. J. “Requirements Engineering: Frameworks for Understanding”. Wiley, 1996.
- IEEE-STD-830-1993. IEEE Recommended Practice for Software Requirements Specifications. Reprinted in Thayer, R. H and Dorfman, M. (eds.) “Software Requirements Engineering, Second Edition”. IEEE Computer Society Press, 1997, p176-205
- MIL-STD-498. Military Standard for Software Development and Documentation. United States Department of Defense. 1994
- S. M. Easterbrook and J. Callahan, “Formal Methods for V&V of partial specifications: An experience report” Third IEEE International Symposium on Requirements Engineering (RE'97), Annapolis, MD., Jan 5-8, 1997.
- Fairley, R. E. and Thayer, R. H. “The Concept of Operations: The bridge from operational requirements to technical specifications”. In Thayer, R. H and Dorfman, M. (eds.) “Software Requirements Engineering, Second Edition”. IEEE Computer Society Press, 1997, p73-83.